AMENDMENTS TO THE CLAIMS

The claims have been amended as follows:

1. (Currently Amended) A method for designing a process, comprising:

(a) designing a model for a process with a visual display surface without developer involvement;

(b) generating a high-level code emission for the process with an association between the model for the process and an user-selected supported, inserted graphical shape-construct corresponding to a visual image; the process being specified by ~~a~~ the visual image on the visual display surface; and

(c) transforming the high-level code emission into computer-executable instructions; through,

determining a first contextual evaluation whether the supported, inserted graphical shape-construct of step (b) is compatible with any previously selected supported, selected graphical shape-construct, and only after having said compatibility is determined,

transforming the association between the model and the high-level code emission for the process is transformed into computer-executable instructions.

2. (Original) The method of claim 1, further comprising:

(d) executing the computer-executable instructions.

3. (Original) The method of claim 1, wherein the process is a business process.

4. (Original) The method of claim 2, wherein (d) is executed on a plurality of computers.

5. (Original) The method of claim 4, wherein the computer-executable instructions are scalable.

6. (Original) The method of claim 2, wherein (d) comprises:

(i) obtaining the high-level code emission;

(ii) retrieving information about an infrastructure on which the computer-executable instructions are executed; and

(iii) executing the computer-executable instructions in accordance with the high-level code emission and the information about the infrastructure,

7. (Original) The method of claim 6, wherein the computer-executable instructions are transport agnostic.

8. (Original) The method of claim 1, wherein (c) comprises:

(i) compiling the high-level code emission to form compiled code; and

(ii) assembling the compiled code to form the computer-executable instructions..

9. (Withdrawn) A method for construct a process, the method comprising:

(a) displaying a visual image to a user, the visual image specifying a process;

(b) receiving a command from the user about the process;

(c) determining whether the command is consistent with semantics of a process type, the process being associated with the process type; and

(d) in response to (c), updating the visual image.

10. (Withdrawn) The method of claim 9, wherein (a) comprises:

(i) supporting a plurality of shapes, each shape corresponding to a construct of the process.

11. (Withdrawn) The method of claim 10, wherein a listen shape corresponds to receiving one of a plurality of message types.

12. (Withdrawn) The method of claim 10, wherein (d) comprises:

(i) in response to (c), positioning a selected shape within the visual image,

13. (Withdrawn) The method of claim 10, wherein (d) comprises:

(i) in response to (c), if the command is not consistent with the semantics, generating an indicator of an error

14. (Withdrawn) The method of claim 10, wherein (d) comprises:

(i) in response to (c), if the command is not consistent with the semantics, rejecting the command.

15. (Withdrawn) The method of claim 13, wherein (d) further comprises:

(ii) positioning the indicator in a proximity of a selected shape.

16. (Withdrawn) The method of claim 15, wherein (d) further comprises:

(iii) generating an explanation of the error.

17. (Withdrawn) The method of claim 16, wherein (d) further comprises:

(iv) positioning the explanation approximately at the proximity of the selected shape.

18. (Withdrawn) The method of claim 10, further comprising:

(e) generating a new construct that is associated with a customization shape.

19. (Withdrawn) The method of claim 18, wherein (e) comprises:

(i) receiving a shape specification of the customization shape;

(ii) receiving a segment of high-level code, the segment being associated with the customization shape; and

(iii) associating the segment with the customization shape.

20. (Withdrawn) The method of claim 9, wherein the command denotes to expand the visual image, and wherein (d) comprises:

(i) increasing a detail of the process by replacing a designated shape with a collection of shapes.

21. (Withdrawn) The method of claim 9, wherein the command denotes to collapse the visual image, and wherein (d) comprises:

(i) reducing a detail of the process by replacing a collection of shapes with a designated shape.

22. (Withdrawn) The method of claim 9, further comprising:

(e) generating a high-level code emission that is consistent with the visual image.

23. (Withdrawn) The method of claim 10, further comprising:

(e) for said each shape, generating at least one line of the high-level code,

24. (Withdrawn) The method of claim 9, wherein the process is a business process.

25. (Withdrawn) A computer-readable medium having computer-executable instructions for performing the method recited in claim 1.

26. (Withdrawn) A computer-readable medium having computer-executable instructions for performing the method recited in claim 6.

27. (Withdrawn) A computer-readable medium having computer-executable instructions for performing the method recited in claim 9,

28. (Withdrawn) A computer-readable medium having computer-executable instructions for performing the method recited in claim 22.

29. (Previously Presented) An apparatus for designing a process, comprising:

a visual designer that displays a visual image and that generates a high-level code emission, the visual image specifying the process and the high-level code emission specifying

the process with an association between the process and the visual image designer that generates the high-level code emission;

a compiler that obtains the high-level code emission from the visual designer and that transforms the high-level code emission into compiled code; through,

a contextual evaluation determining whether selected input to the visual designer is compatible with any previously selected input to the visual designer, and only after having said compatibility determined,

a transformation is executed with the association between the process and the visual image designer generate the high-level code emission into compiled code; and

an assembler that assembles the compiled code into computer-executable instructions.

30. (Original) The apparatus of claim 29, further comprising:

an infrastructure knowledge base that contains information about an infrastructure of an underlying computer system and an execution environment.

a process execution engine that queries the infrastructure knowledge base to obtain the information and executes the computer-executable instructions in accordance with the information, the information being indicative about a configuration of the computer system, the process being executed on the computer system.

31. (Original) The apparatus of claim 30, wherein the process execution engine is remotely located from the visual designer.

32. (Original) The apparatus of claim 30, wherein the process execution engine is co-located with the visual designer.

33. (Original) The apparatus of claim 29, wherein the visual designer comprises:
an input module that receives a command from a user, the command;

a high-level language semantics module that contains semantics of a process type, the process being associated with the process type;

a visual language logic module that determines whether the command is consistent with the semantics of the process type; and

a display module that displays a visual image to a user, that queries the visual language logic module about the command and updates the visual image in accordance with the command, the visual image specifying the process.

34. (Currently Amended) A computer-readable storage medium ~~having~~ storing computer-executable modules, comprising:

(a) an input module that receives a command from a user, the command associated with at least one visual logic module;

(b) a high-level language semantics module that contains semantics of a process type;

(c) an association to the a visual language logic module that determines whether the command is consistent with the semantics of the process type to determine whether the visual language logic is compatible with any previously selected visual language logic, and only after having said compatibility determined will ~~the~~ a query to the visual logic module be initiated; and

(d) a display module that displays a visual image to a user, that queries the visual language logic module about the command and updates the visual image in accordance with the command, the visual image specifying a process, the process being associated with the process type.

35. (Original) The computer-readable medium of claim 34, wherein the display module provides an indication to the user if the command is not consistent with the semantics.

36. (Previously Presented) A method for visually designing a business process, comprising:

(a) displaying a visual image to a user, the visual image specifying a business process;

(b) receiving a command from the user to modify the business process;

(c) determining whether the command is consistent with semantics of a process type, the business process being associated with the process type to determine whether the command is compatible with any previously selected command, and only after having said compatibility determined, will the business process be associated with the process type;

(d) in response to (c), updating the visual image;

(e) in response to (d) representing the process as a high-level code emission; and

(f) transforming the high-level code emission into computer-executable instructions; and

(g) executing the computer-executable instructions in accordance with the high-level code emission and the information about an infrastructure of a system, the business process being executed on the system.

37. (Previously Presented) The method of claim 1, further comprising: allowing the user to bypass any high-level error detection and leaving error detection until the high-level code compiles through a compiler.

38. (Previously Presented) The method of claim 1, further comprising: user-selected customization for inserted graphical shape-constructs, wherein the user may separately add high-level command functions after selecting the user-selected inserted graphical shape-construct.

39. (Previously Presented) The method of claim 1, further is comprising: user-selected customization for inserted graphical shape-constructs, wherein the user may initially enter high-level command functions, wherein the association between the model and the high-level code emission for the process is automatically triggered and a shape-construct is activated according to the high-level code.